VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky

Bakalářská práce

2013 Martin Musialek

VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Vývoj aplikace pro Android OS pro zjišťování informací o mobilní síti

Developing Application for Android OS for Exploring Information About Mobile Network

2013 Martin Musialek

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Martin Musialek**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Vývoj aplikace pro Android OS pro zjišťování informací o mobilní síti
Developing Application for Android OS for Exploring Information About Mobile Network

Zásady pro vypracování:

Cílem bakalářské práce bude vyvinout aplikaci pro zjišťování parametrů mobilních sítí. Bude se jednat konkrétně o parametry sítě GSM a UMTS. Výsledky z měření budou ukládány na lokální úložiště a také budou zobrazovány na vzdáleném počítači. Teoretická část bakalářské práce se bude zabývat popisem aplikace a také teoretickým rozborem jiných způsobů, jakými lze zjistit parametry mobilní sítě na mobilním zařízení.

1. Popište API Google Android OS, zaměřte se na parametry mobílních sítí.
2. Po dohodě s vyučujícím vyberte parametry, které se budou ukládat na SD kartu.
3. Vytvořte aplikaci pro Android OS.
4. Vytvořte dokumentaci k aplikaci.
5. Zobrazování výsledků na mapovém podkladu.

Seznam doporučené odborné literatury:

Podle pokynů vedoucího bakalářské práce.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Lukáš Kapičák**

Datum zadání: 16.11.2012
Datum odevzdání: 07.05.2013

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně.

Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne: 7.5.2013 .....

Podpis

Poděkování

Rád bych poděkoval vedoucímu mé bakalářské práce Ing. Lukášovi Kapičákovi za odbornou pomoc a konzultaci při vytváření této práce.

Abstrakt

Bakalářská práce se zabývá vývojem aplikace pro operační systém Android pro zjišťování parametrů mobilních sítí. Konkrétně se jedná o parametry sítě GSM a UMTS. Výsledky z měření jsou ukládány na lokální úložiště a jsou zobrazovány na vzdáleném počítači. Teoretická část bakalářské práce se zabývá popisem aplikace a také teoretickým rozborem jiných způsobů, jakými lze zjistit parametry mobilní sítě na mobilním zařízení.

Klíčová slova

Android, GSM, UMTS, Mobilní sítě, Google MAPS

Abstract

This thesis deals with the development of applications for Android OS for mobile detection parameters. Specifically, the parameters of the GSM and UMTS networks. The results of the measurements are stored on local storage and are displayed on the remote computer. The theoretical part of the thesis deals with the application description and theoretical analysis of other ways in which parameters can be determined by the mobile network on the mobile device.

Key words

Android, GSM, UMTS, Mobile networks, Google MAPS

Seznam použitých symbolů a zkratek

API	Application Programming Interface
AWT	Abstract Window Toolkit
BSS	Base Station Subsystem
BTS	Base Transceiver Station
CID	Cell ID
DSP	Digital Signal Processing
DVM	Dalvik Virtual Machine
GPRS	General Packet Radio Service
GSM	Global System for Mobile Communications
LAC	Location Area Code
MS	Mobile Station
NSS	Network Switching Subsystem
OS	Operační system
PDA	Personal Digital Assistant
PIN	Personal Identification Number
PUK	Personal Unblocking Key
SD	Secure Digital
SDK	Software Development Kit
SIM	Subscriber Identification Module
SMS	Short Message Service
SQL	Structured Query Language
SSL	Secure Sockets Layer
UMTS	Universal Mobile Telecommunication System

Seznam obrázků

Seznam výpisů zdrojového kódu

Obsah

1 Úvod

Cílem této bakalářské práce bylo vytvořit aplikaci pro zjišťování parametrů mobilní sítě fungující na operačním systému Android OS. Jedná se o parametry sítě GSM a UMTS.

Jelikož operační systém Android OS již překonal 50% zastoupení na trhu s chýtrými telefony, vyplývá z toho, že každý druhý chytrý telefon využívá operační systém Android.[9] V následující kapitole se zabývám stručným popisem tohoto operačního systému.

V třetí kapitole popisuji mobilní síť GSM.

Ve čtvrté kapitole se věnuji mobilní síti třetí generace UMTS.

V páté kapitole popisuji, jaké možnosti pro vývojáře nabízí API Google Android OS pro práci s mobilní sítí a jejími parametry. Najdete zde také stručný popis jiného způsobu zjištění parametrů mobilních sítí.

V šesté kapitole popisuji mojí aplikaci z pohledu vývojáře. Vyskytují se zde ukázky zdrojových kódů a vysvětlení vybraných částí kódů.

Předposlední sedmá kapitola slouží jako uživatelská příručka k aplikaci a v poslední kapitole vyhodnocuji výsledky práce.

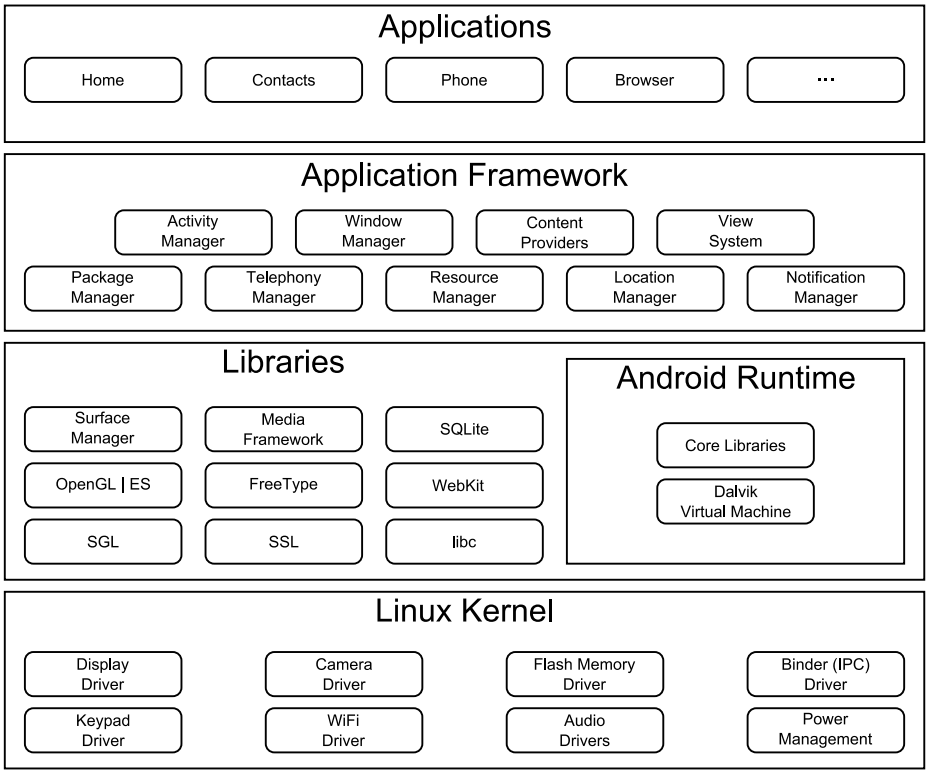
2 Android OS

Operační systém Android měl původně sloužit jako systém pro chytré fotoaparáty. Trh s fotoaparáty ale nebyl tak velký, a proto po převzetí společnosti Android Googlem bylo využití systému přehodnoceno. Soustředili se tedy na vývoj operačního systému pro chytré telefony a tým společnosti Google tak začal vyvíjet konkurenční operační systém pro mobilní verzi operačního systému Windows. V roce 2008 byl na trh uveden první chytrý telefon s operačním systémem Android [8]. V současné době vede operační systém Android na celosvětovém trhu s chytrými telefony a více než 50% všech chytrých telefonů obsahuje operační systém Android [9].

Nástroje a API potřebné k vývoji aplikací pro platformu Android jsou obsaženy v Android SDK. Pro vývoj se používá programovací jazyk Java, a komu by nevyhovovaly standardní knihovny, může si vytvořit svou vlastní knihovnu v programovacím jazyce C/C++.

2.1 Architektura

Architektura operačního systému Android je rozdělena do 5 vrstev. Každá vrstva má svůj účel a nemusí být přímo oddělena od vrstev ostatních. Rozdělení komponent do jednotlivých vrstev si můžete prohlédnout na následujícím obrázku.



Obrázek 1 Architektura operačního systému Android

Nejnižžší vrstva označená *Linux Kernel* tvoří jádro operačního systému Android. Jádro systému Android tvoří abstraktní vrstvu mezi používaným hardwarem a zbytkem softwaru ve vyšších vrstvách. *Linux* je využit pro základní systémové služby jako je bezpečnost, správa paměti, správa procesů, síťové prostředky a zabudované ovladače. Důvodem použití jádra *Linux* byla také vlastnost poměrně snadného použití na různých zařízeních a tím zaručená přenositelnost [4]. Aplikace pro Android jsou napsány v jazyce Java. Nástroje obsažené v Android SDK zkompilují kód zároveň s daty a zdroji do jednoho balíku, kterým je archiv s koncovkou .apk. Tento soubor se považuje za celou aplikaci a slouží k instalaci do zařízení.

Další vrstva je označena *Libraries*. Vrstva *Libraries* obsahuje knihovny, které jsou napsány v programovacím jazyce C/C++ a jsou využívány různými komponentami operačního systému Android. Funkce, které tyto knihovny nabízí jsou vývojářům poskytnuty prostřednictvím vyšší vrstvy *Android Application Framework*. Touto částí se budu zabývat později. Nyní uvedu některé příklady knihoven:

- **Media Framework** – Knihovna podporující multimédia.
- **WebKit** – Knihovna webového prohlížeče.
- **libc** – Standardní knihovna jazyka C.
- **SQLite** – Knihovna pro práci s databázemi.
- **SSL** – Knihovna pro zabezpečení datových přenosů.
- **FreeType** – Knihovna pro vykreslování fontů.
- **OpenGL** – Knihovna pro vykreslování grafiky.

V praktické části mé bakalářské práce jsem použil následující knihovny:

- **android.app** – Knihovna poskytující přístup k aplikačnímu modelu a obsahující API funkce aktivit a služeb, které tvoří základ pro všechny Android aplikace.
- **android.content** – Knihovna obsahující třídy pro přístup a publikování dat v zařízení.
- **android.location** – Knihovna poskytující přístup k aktuální fyzické poloze zařízení za použití lokalizace přes GPS přijímač.
- **android.telephony** – Knihovna umožnující komunikovat se zařízením přímo a sledovat tak stav telefonu.
- **android.text** – Knihovna obsahující nástroje pro zpracování a analýzu řetězců a jejich zobrazení.
- **android.widget** – Knihovna umožňující přístup k prvkům uživatelského rozhraní, jako jsou tlačítka, seznamy, atd.
- **com.google.android.maps** – Balíček knihoven umožňující vývojáři přístup k aplikaci Google Maps.

Následující vrstva *Android Runtime* obsahuje virtuální stroj DVM (Dalvik Virtual Machine). Virtuální stroj Dalvik byl vyvíjen od roku 2005 speciálně pro Android společností Google pod vedením Dana Bornsteina. DVM má registrově orientovanou architekturu a využívá základních vlastností Linuxového jádra, jako je například správa paměti nebo práce s vlákny. V této vrstvě jsou také obsaženy základní knihovny programovacího jazyka Java. Knihovny se svým obsahem blíží platformě Java Standard Edition. Hlavní rozdíl je v nepřítomnosti knihoven pro uživatelské rozhraní (AWT a Swing), které byly nahrazeny knihovnami uživatelského rozhraní pro Android a přibyly také knihovny Apache pro

práci se sítí. Překlad aplikace napsané pro Android probíhá zkompilováním zdrojového Java kódu do Java byte kódu pomocí stejného kompilátoru, který je používán v případě překladu Java aplikací. Poté se překompiluje Java byte kód pomocí Dalvik kompilátoru a výsledný Dalvik byte kód je spuštěn na DVM. Každá spuštěná Android aplikace je spuštěna ve svém vlastním procesu, s vlastní instancí DVM [4] [10].

Čtvrtou vrstvou je *Application Framework*. Tato vrstva je z pohledu vývojáře, dá se říct, nejdůležitější. *Application Framework* umožňuje vývojářům přistoupit ke službám, které mohou použít přímo ve svých aplikacích, a ty vývojářům následně dovolí například přistoupit k prvkům uživatelského rozhraní, používat hardware zařízení, nastavovat alarmy atd. Mezi nejdůležitější služby, které nabízí *Application Framework* patří:

- **Prvky View** – Tyto prvky jsou použity pro sestavení uživatelského rozhraní jako seznamy, textové pole, tlačítka, checkboxy a jiné.
- **Content Providers** – Umožňuje přístup k obsahu jiných aplikací.
- **Resource Manager** – Poskytuje přístup k „nekódovým“ zdrojům, jako jsou řetězce, grafika, přidané soubory.
- **Notification Manager** – Umožňuje všem aplikacím zobrazit vlastní upozornění ve stavovém řádku.
- **Activity Manager** – Ovládá životní cyklus aplikací, jejich start, průběh a ukončení.
- **Package Manager** – Obsahuje informace o aplikacích nahraných do operačního systému.

V mé aplikaci *GSMInformator*, kterou se budu zabývat později, využívám především služby:

- **Telephony Manager** – Umožňuje přístup k informacím o telefonních službách.
- **Location manager –** Pravidelně poskytuje aktuální informace o geografické poloze zařízení, pokud je aktivován GPS přijímač.

Poslední a zároveň nejvyšší vrstvu operačního systému Android tvoří *Applications.* Tato vrstva obsahuje základní aplikace, které využívají běžní uživatelé. Způsobů, jakými lze aplikace získat je hned několik. Některé základní aplikace už může mít uživatel předinstalovány ve svém zařízení. Dalším způsobem je zakoupení aplikací prostřednictvím oficiální distribuční služby *Google Play Store.* Na *Google Play Store* je také k dispozici spousta aplikací zdarma. Před zakoupením, nebo stažením aplikace však doporučuji přečtení uživatelských recenzí. Zkušenější uživatelé si mohou aplikace naprogramovat sami prostřednictvím programovacího jazyka Java.

2.2 Základní části aplikace Android

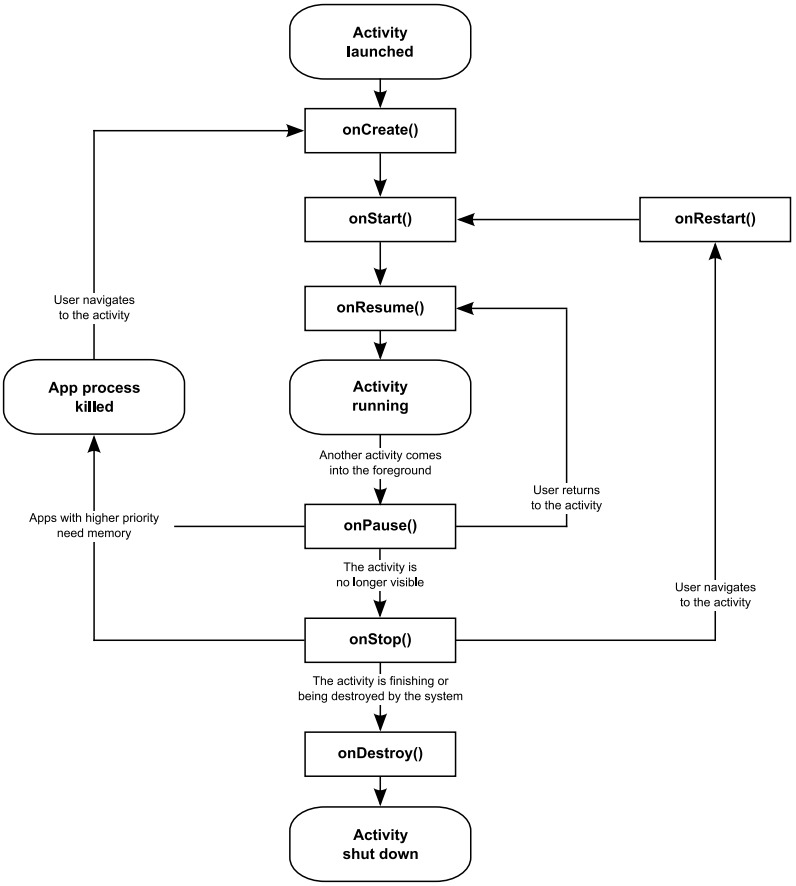
Mezi základní stavební kameny v aplikacích Android patří komponenty:

- **Activity** - Představuje uživatelské rozhraní umožňující interakci s uživatelem.
- **Service** - Umožňuje provádět akce na pozadí.
- **Content Providers** - Poskytuje přístup k datům.
- **Broadcast Reciever** - Reaguje na příchozí oznámení.

Všechny tyto komponenty musí být definovány v souboru *AndroidManifest.xml*, uloženém v kořenovém adresáři projektu. Kromě *Content Providers* mohou komponenty navzájem spolupracovat pomocí intents. *Intents* neboli *záměry* jsou definovány jako abstrakce operace, kterou je potřeba vykonat. *Záměr* se obecně skládá z činnosti, která se má vykonat, parametru, nad kterým má být tato činnost vykonána, a aplikace, která má tuto činnost provést [4].

2.2.1 Activity

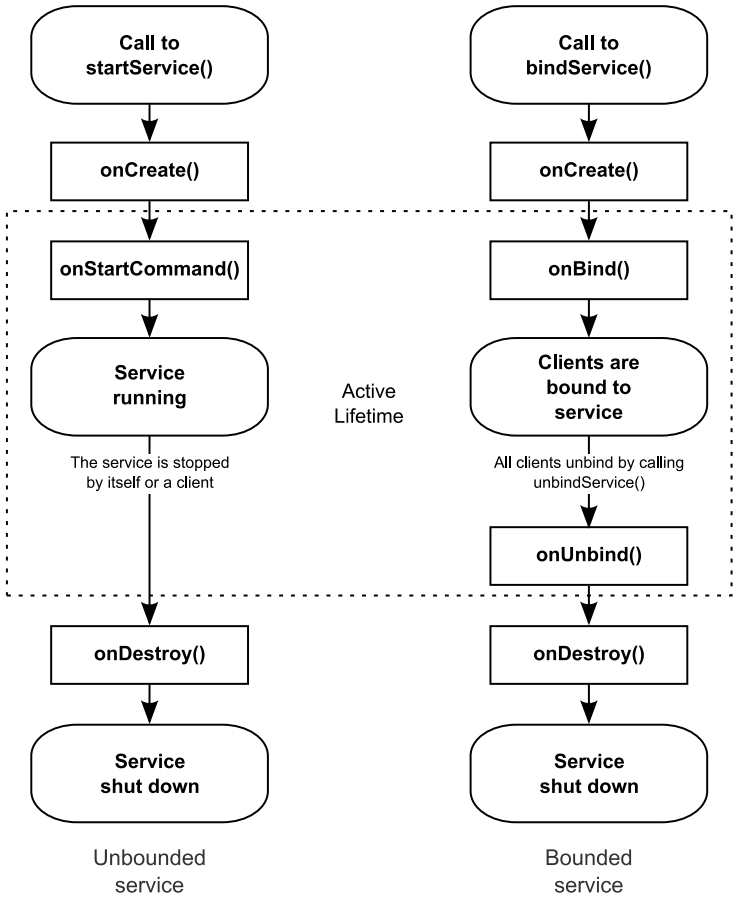
Aktivita odpovídá jedné obrazovce. Obsahuje grafické uživatelské rozhraní pro interakci s uživatelem. Aplikace obsahuje obvykle více *aktivit*, mezi kterými je uživatel schopen přepínat a přitom si *aktivity* mohou předávat informace. Zahájení *aktivity* je poměrně náročná záležitost. Musí se vytvořit nový proces, alokovat paměť pro objekty uživatelských rozhraní, které se rozloží do layoutu obrazovky a připravenou obrazovku zobrazit. Aby nedocházelo ke zbytečnému plýtvání výpočetních prostředků např. při vzniku, zániku a opětovného vzniku *aktivity* existuje *Activity Manager*, který zodpovídá za vytváření, rušení a celkovou správu životního cyklu *aktivity*. *Activity Manager* pracuje se zásobníkem, ve kterém jsou uchovány informace o spuštěných *aktivitách*, a na vrcholu tohoto zásobníku je aktuálně zobrazovaná *aktivita* [4] [5]. Životní cyklus *aktivit* je znázorněn na obrázku č. 2.



Obrázek 2 Životní cyklus aktivity

2.2.2 Service

Komponenta *service* neboli *služba* neposkytuje uživatelské rozhraní. Většinou se používá k vykonávání dlouho trvajících úkolů nebo k přístupu ke vzdáleným zdrojům, kde není známá doba odezvy (jako je připojení k serveru). *Službu* můžeme spustit dvěma způsoby, a to pomocí metody *startService* nebo *bindService*. Po spuštění metodou *startService* se *služba* může ukončit sama nebo ji může ukončit jiná komponenta. Životní cyklus *služby* spuštěné metodou *startService* můžete vidět na obrázku č. 3 vlevo. Po spuštění *služby* pomocí metody *bindService*, kterou vyvolá jiná komponenta, tzv. klient, může *službu* ukončit pouze klient, který ji spustil. V jednom okamžiku může být ke *službě* navázáno pomocí metody *bindService* i více komponent, potom je *služba* ukončena po odpojení všech klientů. Životní cyklus *služby* spuštěné metodou *bindService* můžete vidět na obrázku č. 3 vpravo [4][6].



Obrázek 3 Životní cyklus služby

2.2.3 Content provider

Content provider je aplikační rozhraní pro sdílení dat mezi aplikacemi, ale i pro sdílení dat v aplikaci mezi jednotlivými *aktivitami*. Aplikace může uchovávat data v souborech, SQLite databázi nebo na webu, a přesto budou mít k těmto datům přístup, pokud je to povoleno, jiné aplikace. *Content provider* má relativně jednoduché rozhraní se standardními metodami (*insert*, *update*, *delete* a *query*), které mají stejnou funkci jako klasické databázové metody. Oddělení dat od uživatelského rozhraní nabízí

možnost nahrazení výchozích aplikací novými. Například může jakákoliv aplikace využít uložených uživatelských kontaktů a nahradit tak výchozí aplikaci pro jejich zobrazování [4][7].

2.2.4 Broadcast receiver

Broadcast reciever je komponenta sloužící k „naslouchání“ oznámení. Podle určení na ně reaguje, například výpisem na stavový řádek nebo spuštěním jiné komponenty. Aplikace mohou využívat *broadcasty* systémové nebo vytvářet své vlastní. Podobně jako *služba* ani *broadcast receiver* nemá uživatelské rozhraní. Příklad použití *broadcast recieveru* může být reakce na oznámení o nízkém stavu baterie, o zachycení fotografie, doručení SMS zprávy nebo stažení dat [11].

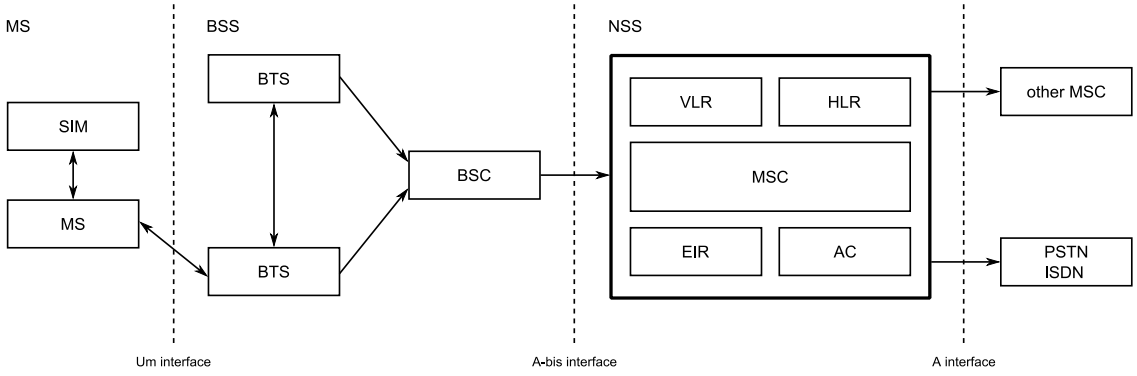
3 GSM

Mobilní síť GSM patří mezi standardy druhé generace mobilních sítí. GSM využívá digitální přenos dat, který zajišťuje kvalitnější spojení v nepříznivých podmínkách pozemních rádiových kanálů. Přenos signálů v digitální formě umožňuje značně rozšířit nabídku poskytovaných služeb a dosáhnout tak kompatibility s jinými digitálními sítěmi, a to nejen v rámci jednoho státu, ale i po celém světě. GSM používá pro pokrytí území rádiovým signálem celulární(buňkovou) síť. Požadované území, například území jednoho státu, je rozděleno na velké množství buněk. Uprostřed každé buňky je obvykle umístěna základnová stanice BTS, která zajišťuje spojení mobilních stanic se systémem [13].

3.1 Struktura sítě GSM

Architekturu mobilní sítě GSM rozdělujeme na 3 subsystémy. Jedná se o:

- BSS (Base Station Subsystem) – Subsystém základnových stanic
- NSS (Network and Switching Subsystem) – Síťový a spínací subsystém
- OSS (Operation Support Subsystem) – Operační subsystém



Obrázek 4 Architektura sítě GSM

3.1.1 Mobilní stanice (Mobile station MS)

Dle specifikací GSM se mobilní stanicí rozumí vlastní mobilní zařízení (vysílač/přijímač) a také předplatitelský identifikační modul SIM (Subscriber Identification Module), který umožňuje unikátní identifikaci uživatele v rámci celé sítě GSM.

SIM karta – je součástí mobilní stanice a obsahuje informace o majiteli karty, čtyřmístné identifikační číslo PIN (Personal Identification Number) a neměnné identifikační číslo PUK (Personal Unblocking Key). Hlavním smyslem SIM karty je ověření a identifikace uživatele a služeb jemu přístupných. SIM karta je přenosná a lze ji použít s libovolným mobilním telefonem [12].

SIM karta dále uchovává následující údaje :

- IMSI (International Mobile Subscriber Identity)
- Autentikační klíč
- Šifrovací klíč
- TMSI (Temporary Mobile Subscriber Identity)
- LAI (Location Area Identity)

Mobilní telefon obsahuje rádiový přijímač/vysílač, pomocí kterého komunikuje se základnovými stanicemi. Mobilní zařízení je jednoznačně identifikováno pomocí IMEI (International Mobile Equipment Identity) čísla [2] [3] [12].

3.1.2 Subsystém základnových stanic (BSS)

Subsystém BSS se skládá z:

- Základnové stanice (BTS - Base Transceiver Station)
- Základnové řídící jednotky (BSC - Base Station Controller)
- Transkodéru (TC - TransCoder)

Základnové stanice BTS komunikují prostřednictvím rádiového rozhraní *Um* s jednotlivými mobilními stanicemi MS. BSC vytváří komunikační spojnici mezi MS a MSC a transkóduje 13kbps hlasový kanál do standardního 64kbps kanálu (PSTN, ISDN) [1] [13].

3.1.3 Síťový subsystém (NSS)

Základní funkce NSS jsou: registrace v síti, ověřování, lokalizace polohy, směrování hovorů, roaming a spojení mezi pevnou sítí. Hlavní komponentou NSS je *mobilní spínací ústředna* (MSC), která zajišťuje funkci telefonní ústředny. Mezi další komponenty síťového subsystému NSS patří:

- *Domovský lokační registr* (HLR - Home Location Register) – databáze, která obsahuje všechny důležité informace o uživatelích
- *Návštěvnický lokační registr* (VLR - Visitor Loaction Register) - obsahuje vybrané informace z HLR, nezbytné pro řízení hovorů těch mobilních stanic, které se právě pohybují v dané geografické oblasti spravované danou MSC.
- *Registr mobilních stanic* (EIR - Equipment Identity Register) - databáze, která obsahuje seznam všech platných mobilních telefonů celé sítě, kde je každý účastník identifikován pomocí IMEI čísla.
- *Autentifikační centrum* (AuC - Authentification Center) - je chráněná databáze, která obsahuje kopii tajných klíčů, které jsou uloženy na SIM kartě, a které se používají při přihlášení do sítě [13].

3.1.4 Operační subsystém (OSS)

OSS má na starost provoz a údržbu celého systému. Mezi základní funkce toho subsystému patří:

- Podíl na managementu mobilních stanic.
- Dohled nad síťí, a také její konfigurace.
- Kontrola a údržba systému GSM.
- Podíl na administrativním managementu účastníků GSM, konkrétně na jejich registraci.

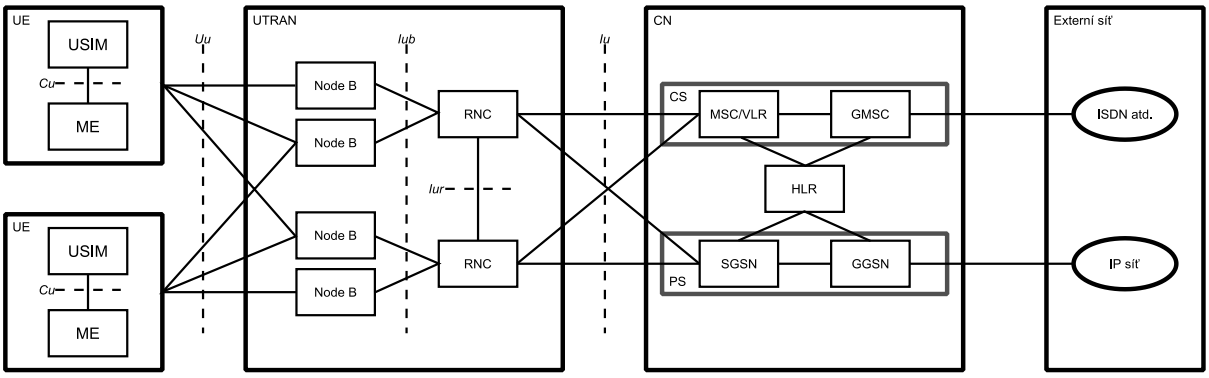
4 UMTS

UMTS patří mezi mobilní sítě třetí generace, které jsou primárně zaměřeny na multimediální služby a vysokorychlostní datové přenosy. V sítích UMTS je na rádiovém rozhraní použita varianta WCDMA (Wideband Code Division Multiple Access) se šířkou pásma 5 MHz [13].

4.1 Struktura sítě UMTS

UMTS síť lze rozdělit na tři části:

- Uživatelská stanice (UE – User Equipment)
- Síť rádiového přístupu (UTRAN - Universal Terrestrial Radio Access Network)
- Páteřní síť (CN – Core Network)



Obrázek 5 Architektura sítě UMTS

4.1.1 UE – Uživatelská stanice

UE umožňuje koncovým uživatelům přistupovat k UMTS síti. Uživatelská stanice představuje mobilní telefon nebo UMTS modem a rozdělujeme ji na dvě části:

- ME (Mobile Equipment) – slouží k navázání rádiového spojení přes rádiové rozhraní
- USIM (UMTS Subscriber Identity Module) – slouží k identifikaci uživatele

4.1.2 UTRAN – Obecná rádiová přístupová síť

UTRAN představuje část sítě, která umožňuje uživatelům mobilní přístup ke službám poskytovaným páteřní sítí CN pomocí rádiového rozhraní *Uu*. UTRAN zprostředkovává rádiový přenos. Plní také funkci řízení a přidělování rádiových prostředků.

V systému UTRAN jsou definovány dvě síťové jednotky:

- Node B
  - jedná se o základnovou stanici systému UMTS (obdoba BTS v GSM)
  - obsahuje rádiové přijímače/vysílače a anténní systém obsluhující jednu nebo více buněk
  - základními funkcemi síťové jednotky Node-B jsou modulace a demodulace, vysílání a příjem, kódování fyzických kanálů, ochrana proti chybám a řízení vysílacího výkonu

- RNC (Radio Network Controller)
  - řídící jednotka rádiové sítě (obdoba BSC v GSM)
  - kontroluje funkčnost jedné nebo několika základnových stanic
  - stará se o šifrování a řízení handoveru
  - řídí vysílací výkon

Síť UTRAN se skládá z jednoho nebo více subsystémů rádiových sítí RNS (Radio Network Subsystem). Každý subsystém RNS je řízen jednotkou RNC, která je připojena do páteřní sítě prostřednictvím rozhranní *Iu*. RNC jsou mezi sebou propojeny přes rozhraní *Iur*. RNC má na starost rádiové zdroje a jejich management pro určitou geografickou oblast, rozdělenou na jednotlivé buňky. O pokrytí jednotlivých buněk rádiovým signálem se starají základnové stanice Node B. Základnové stanice jsou s RNC propojeny přes rozhranní *Iub* [13].

4.1.3 CN – Jádro sítě

Jádro sítě CN provádí spojovací funkce, kterými jsou propojení účastníků a směrování paketů. Dále udržuje a aktualizuje důležité uživatelské informace a zajišťuje spojení do dalších sítí [13].

Jádro sítě UMTS je rozděleno na dvě domény:

- Okruhově spínaná doména (CS – Circuit Switched) – používá přepojování okruhů k přenosu telefonních hovorů, SMS a dat s využitím techniky přepojování okruhů
- Paketově spínaná doména (PS – Packet Switched) – používá přepojování paketů k přenosu dat a SMS

5 Možnosti získání parametrů mobilních sítí

5.1 API Google Android OS

API operačního systému Android nabízí velké množství tříd a funkcí, které jsou velmi užitečné při vývoji jakékoli aplikace pro Android OS. Já jsem v praktické části mé bakalářské práce využil při vývoji aplikace *GSMInformator* balíček knihoven *android.telephony*.

V následujícím textu můžete nalézt některé parametry, které lze za pomocí balíčku knihoven *android.telephony* zjistit:

- MCC – kód země
- MNC – kód sítě
- CID – identifikační číslo buňky
- LAC – kód území
- Síla signálu sítě
- Název operátora
- Kód operátora
- IMEI – identifikační kód mobilního telefonu
- Typ telefonu
- Typ sítě
- Sériové číslo SIM karty
- Stav SIM karty
- IMSI – identifikační kód určený operátorem pro SIM kartu

Po konzultaci s vedoucím mé bakalářské práce panem Ing. Lukášem Kapičákem, jsem si vybral tyto parametry:

- CID – identifikační číslo buňky
- LAC – kód území
- Síla signálu sítě
- Název operátora

Do aplikace by samozřejmě šly zahrnou všechny parametry, ale pro účely aplikace budou tyto čtyři parametry stačit. K jejich získání jsem z balíčku knihoven *android.telephony* použil následující třídy:

- **TelephonyManager** – Poskytuje přístup k telefonním službám v zařízení.

- **PhoneStateListener** – Umožňuje sledování změn konkrétních stavů telefonního zařízení, např. provozní stav, síla signálu atd.
- **SignalStrength** – Obsahuje informace o síle telefonního signálu.
- **NeighboringCellInfo** – Představuje informace o sousední buňce včetně síly signálu a identifikačního čísla buňky

Další balíček, který jsem využil, nese název *android.location.* Tento balíček slouží k určení polohy a zjištění potřebných parametrů. Konkrétně se jedná o zeměpisnou délku a šířku. Pomocí těchto parametrů pak lze zařízení zobrazit na mapovém podkladu. K získání zeměpisné šířky a délky jsem využil třídy *Location*, kde jsem pomocí metod *getLatitude* a *getLongitude* tyto parametry získal.

5.2 RIL – Radio Interface Layer

Mimo použití API operačního systému Android, je zde také možnost využití nižších vrstev operačního systému k zjištění parametrů mobilních sítí. *RIL* je abstraktní vrstva, která umožňuje komunikaci mezi telefonními službami Androidu (*android.telephony*) a *hardwarem*. *RIL* se skládá ze dvou hlavních částí:

- RIL Daemon – část Androidu
    - Inicializuje Vendor RIL
    - Zpracovává veškerou komunikaci systému Android jako vyžádané příkazy (hovory, odeslání SMS atd.)
- Vendor RIL - část, kterou vytváří výrobce mobilního zařízení
    - Inicializuje RIL Daemon
    - Zpracovává veškerou komunikaci s *hardwarem* a vrací *RIL Daemonu* nevyžádané odpovědí (příchozí hovor, síla signálu sítě atd.)

Pro práci v této vrstvě se používá programovací jazyk C++ [14].

6 GSMInformator

V této kapitole můžete nalézt informace o aplikaci *GSMInformator* a ukázky zdrojových kódů. V podkapitole 6.1 je vysvětlen zakladní princip aplikace. V podkapitole 6.2 se zabývám strukturou aplikace.

6.1 Seznámení s aplikací

Aplikaci, která je výstupem této bakalářské práce, jsem nazval GSMInformator. Aplikace je vyvinuta pro mobilní zařízení s operačním systémem Android OS a byla testována na mobilním zařízení značky HTC, konkrétně se jednalo o model Sensation s operačním systémem Android OS ve verzi 4.0.3. Jelikož mají být výsledky měření z aplikace zobrazovány na vzdáleném počítači, vyvinul jsem i uživatelské rozhraní v programovacím jazyce C# pro uživatele vlastnící počítač s operačním systémem Windows. Aplikace na Android OS funguje jako služba, která zapisuje parametry sítě GSM na SD kartu mobilního zařízení jako klasický textový soubor. V uživatelském rozhraní má pak uživatel možnost si zobrazit aktuální měřená data a výsledky na mapovém podkladu. V aplikaci lze také nahlížet do historie měřených dat, kde si může uživatel vybrat jím požadované datum a čas předchozího měření a výsledky si zobrazit. Aktuální měřená data lze odeslat v XML souboru na vzdálený server, kde je soubor uložen. Tento soubor využívá aplikace na operačním systému Windows jako databázi měřených dat pro jejich následné zobrazení. Obě aplikace jak pro operační systém Android, tak pro operační systém Windows tedy pro svou synchronizaci potřebují připojení k internetu.

6.2 Struktura aplikace

V příloze A si můžete prohlédnout obrázek, na kterém je znázorněna adresářová struktura aplikace.

Popis důležitých složek:

- **src** – V tomto adresáři jsou zdrojové kódy.
- **gen** – Tento adresář obsahuje automaticky generované zdrojové kódy.
- **assets** – Adresář s přídavnými soubory.
- **res** – Zde jsou obsažena zdrojová data(řetězce, grafika atd.), která jsou důležitá k provozu aplikace a jsou pouze ke čtení. Tyto data jsou většinou v XML formátu.
- **bin** – Tento adresář obsahuje přeložené binární soubory .

Mezi nejdůležitější soubory aplikace patří *AndroidManifest.xml*. Tento soubor obsahuje specifikace aplikace, jako je název nebo verze. Obsahuje také specifikace jednotlivých komponent aplikace - aktivit, služeb, content provideru a broadcast receiverů.

Pro správnou funkčnost mé aplikace jsem do souboru *AndroidManifest.xml* musel přidat následující řádky, kterými definuji práva, která bude aplikace mít.

Výpis 6-1 Nastavení oprávnění v AndroidManifest.xml

```
<uses-permission
     android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission
     android:name="android.permission.INTERNET" />
<uses-permission
     android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission
     android:name="android.permission.READ_PHONE_STATE" />
<uses-permission
     android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission
     android:name="android.permission.ACCESS_NETWORK_STATE" />
```

Význam jednotlivých práv:

- **WRITE_EXTERNAL_STORAGE** – umožňuje aplikaci zapisovat na externí úložiště
- **INTERNET** – umožňuje aplikaci otevřít síťové sockety
- **ACCESS_COARSE_LOCATION** – umožňuje aplikaci získat přibližnou polohu odvozenou z GSM věží nebo Wi-Fi
- **READ_PHONE_STATE** – umožňuje aplikaci číst stav telefonu
- **ACCESS_FINE_LOCATION** - umožňuje aplikaci získat přesnou polohu odvozenou z GPS
- **ACCESS_NETWORK_SATE** – umožňuje aplikaci přístup k informacím o síti

V následujících podkapitolách můžete nalézt některé třídy a metody, které jsem ve své aplikaci *GSMInformator* použil a ukázky ze zdrojových kódů.

6.2.1 Třída MainServiceActivity

Třída *MainServiceActivity.java* je hlavní třídou aplikace. Při prvním spuštění aktivity se jako první spouští metoda *onCreate*, která se postará o zobrazení uživatelského rozhraní a zaregistrují se obsluhy událostí. Spustí se zde také metoda *downloadBTSList*, která provádí stažení souboru *btslist.xml* z FTP serveru na SD kartu mobilního zařízení.

Výpis 6-2 Metoda *downloadBTSList*

```
public void downloadBTSList()
{
    //Download from FTP server
    new Thread(new Runnable()
```

```
    {
          public void run() {

             FTPClient client = new FTPClient();
             try
             {
                  client.connect("ftp.martinmusialek.php5.cz");
                  client.login("USERNAME", "PASS");
                  String filename = "btslist.xml";
                  File file = new File("/sdcard/btslist.xml");
                  fos = new FileOutputStream(file);
                  boolean result = client.retrieveFile(filename,
fos);
                  fos.close();
                 client.logout();
                client.disconnect();
             }
             catch (IOException e)
             {
                  e.printStackTrace();
             }
          }
    }).start()
}
```

Soubor *btslist.xml* jsem vytvořil jako databázi základnových stanic BTS, u každé stanice uchovávám jejich CID, adresu a GPS souřadnice, kde se daná stanice nachází. Tyto informace jsem získal na internetové adrese www.gsmweb.cz, kde je volně dostupný seznam BTS stanic. Jelikož mi ale žádný z dostupných seznamů nevyhovoval, vytvořil jsem si vlastní seznam. Dále se v nových vláknech spouští metody *readBTSList* a *readGSMReport*. Obě metody si jsou velice podobné, s tím rozdílem, že zracovávají jiné soubory. Metoda *readGSMReport* čte soubor *GSMReport.xml*, který tvoří sám uživatel aplikace, když nahrává své výsledky měření na server. Reportují se v něm všechny měřené parametry včetně času měření.

Níže na výpisu 6-3 můžete vidět způsob čtení souboru. Jedná se konkrétně o čtení souboru *GSMReport.xml*.

Výpis 6-3 Čtení souboru GSMReport.xml

```
xpp.nextTag();
xpp.require(XmlPullParser.START_TAG, null, "GSMInformator");
while (xpp.nextTag() == XmlPullParser.START_TAG)
{
```

```
        xpp.require(XmlPullParser.START_TAG, null, "report");
        xpp.nextTag();
        xpp.require(XmlPullParser.START_TAG, null, "date");
        String date = xpp.nextText();
        xpp.require(XmlPullParser.END_TAG, null, "date");
        xpp.nextTag();
        xpp.require(XmlPullParser.START_TAG, null, "lac");
        String lac = xpp.nextText();
        xpp.require(XmlPullParser.END_TAG, null, "lac");
        .
        .
        .
        xpp.nextTag();
        xpp.require(XmlPullParser.END_TAG, null, "report");

        GSMReport thisGSMReport = new GSMReport(
        date, lac, cidd, signal, operator, la, lo, neighbouringCells);
        gsmReport.add(thisGSMReport);
}
xpp.require(XmlPullParser.END_TAG, null, "GSMInformator");
```

Dále se v metodě *onCreate* vytváří *PhoneStateListener* k získání potřebných parametrů mobilní sítě a také *Location* posluchač k získání GPS pozice.

Je zde také zaregistrována obsluha událostí, konkrétně události kliknutí na tlačítka. Tlačítko *View Map* spouští novou aktivitu *MyGoogleMap*. Informace o této aktivitě naleznete v podkapitole 6.2.3. Po stisknutí tlačítka *Upload Report* se nejdříve uloží aktuální data do souboru *GSMReport.xml*, který se následně odešle na server.

Další používanou metodou je *haveNetworkConnection*, která je typu boolean a pouze zkoumá, zda je mobilní zařízení připojeno k internetu ať už prostřednictvím mobilní sítě nebo bezdrátové technologie Wi-Fi. Pokud je zařízení připojeno k internetu, vrací hodnotu true a pokud ne, vrátí hodnotu false.

Výpis 6-4 Metoda haveInternetConnection

```
private boolean haveNetworkConnection()
{
        boolean haveConnectedWifi = false;
        boolean haveConnectedMobile = false;
        ConnectivityManager cm =  (ConnectivityManag-
er)getSystemService(Context.CONNECTIVITY_SERVICE);
```

```
      NetworkInfo[] netInfo = cm.getAllNetworkInfo();
      for (NetworkInfo ni : netInfo)
      {
            if (ni.getTypeName().equalsIgnoreCase("WIFI"))
               if (ni.isConnected())
                  haveConnectedWifi = true;
            if (ni.getTypeName().equalsIgnoreCase("MOBILE"))
               if (ni.isConnected())
                  haveConnectedMobile = true;
      }
      return haveConnectedWifi || haveConnectedMobile;
}
```

6.2.2 Třída FirstService

Aby aplikace mohla být spuštěna jako služba, je zapotřebí této třídy *FirstService*. Tato služba provádí neustálé měření parametrů mobilní sítě, a ty pak zapisuje do souboru *GSMReport.txt*. Soubor je následně uložen na paměťovou kartu mobilního zařízení. Data se v souboru nepřepisují, ale přidávají se nové hodnoty. Velikost souboru tedy neustále roste. Jedná se ale o připisování poměrně malého množství dat, a tak je velikost souboru zanedbatelná vzhledem k velikostem kapacit dnešních paměťových karet. Zápis dat probíhá jednoduše pomocí metod třídy *Writer*. Aby se data stále připisovala je pro zápis použita metoda *append*.

Výpis 6-5 Zápis parametrů do souboru

```
File myFile = new File("/sdcard/GSMReport.txt");
Time now = new Time(Time.getCurrentTimezone());
now.setToNow();
Writer fw = new BufferedWriter(new FileWriter(myFile, true));
fw.append("GSM parameters: \nTime: … ");
.
.
.
fw.close();
```

6.2.3 Třída MyGoogleMap

Zde popisuji výše uvedenou třídu *MyGoogleMap*, která slouží pro zobrazení výsledků na mapovém podkladu. Jedná se o klasickou třídu, která dědí metody z třídy *MapActivity*. Nejedná se ale o standardní balík knihovny Android, proto bylo zapotřebí přidat do souboru *AndroidManifest.xml* následující řádek:

Výpis 6-6 Přidání knihovny v AndroidManifest.xml

```
<uses-library android:name="com.google.android.maps" />
```

Výpis 6-6 říká, že aplikace bude používat připojenou knihovnu s názvem uvedeným v atributu *name*. V tomto případě se tedy jedná o Google Maps API pro Android OS.

Po přidání této knihovny jsem mohl ve své aplikaci pracovat s Google Mapami. Na mapě se zobrazuje aktuální poloha zařízení a poloha okolních BTS stanic. Pokud není aktivován GPS přijímač, zobrazí se pouze okolní stanice. Pro zobrazení objektů na mapě je implementována třída *Overlay*, ze které musí objekt dědit, aby mohl být na mapě zobrazen. Následně lze z třídy *MapView* pomocí metody *getOverlays* získat seznam *List<Overlay>*, který reprezentuje všechny objekty zobrazené na mapě.

6.2.4 Ostatní třídy

Třída *HistoryActivity* představuje aktivitu, která zobrazuje informace z předchozích měření. Zobrazení dat probíhá za pomocí třídy *Spinner*.

Výpis 6-7 Naplnění spinneru daty

```
Spinner spinner = (Spinner)findViewById(R.id.spinner1);
array_spinner = new String[dateList.size()];

for(int i = 0 ; i < dateList.size() ; i++)
{
     array_spinner[i] = dateList.get(i).getDate();
}
ArrayAdapter<String> adapter =
new ArrayAdapter<String>(this, android.R.layout.simple_spinner_item,
array_spinner);
spinner.setAdapter(adapter);
```

Smyčkou *for* se prochází seznam *dateList*, který obsahuje všechny informace o datech měření, a ty se pak přidávají do pole *array_spinner*. Toto pole je pak předáno jako parameter při vytvoření datového adaptéru. Nakonec už jen stačí nad objektem *spinner* zavolat metodu *setAdapter* a jako parameter jí předat vytvořený datový adaptér. Nyní je *spinner* naplněn daty a čeká se na událost výběru položky *onItemSelected*. Jakmile je položka vybrána, nastaví se podle data všechny *TextView*.

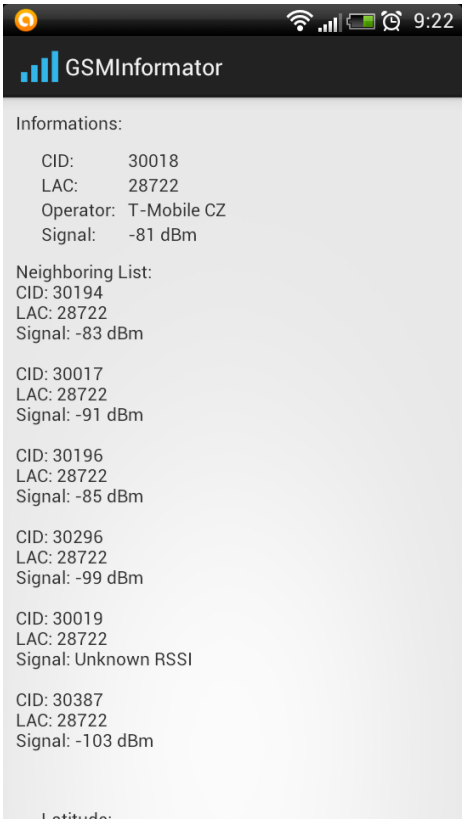
Další třída *MyGoogleMapHistory* představuje aktivitu, která funguje stejně jako *MyGoogleMap* s tím rozdílem, že *MyGoogleMapHistory* zobrazuje objekty na základě dat zobrazených v *HistoryActivity* a nikoli aktuálních dat.

7 Uživatelský manuál

Tato kapitola slouží jako uživatelský manuál k aplikaci. Podrobně zde popisuji veškeré funkce aplikace, které může uživatel při svém měření využít. Zabývám se zde take aplikací *GSMInformator* pro operační systém Windows, kterou jsem musel vytvořit pro možnost zobrazení výsledků měření z aplikace mobilního zařízení na vzdáleném počítači.

7.1 Aplikace pro Android OS

Při spuštění aplikace jsou na úvodní obrazovce měřená data. V horní části *Informations* se nacházejí informace o aktuální buňce. Naleznete zde identifikační číslo buňky CID, kód území LAC, název operátora a sílu signálu mobilní sítě. V další části *Neighboring List* jsou informace o dostupných okolních buňkách.



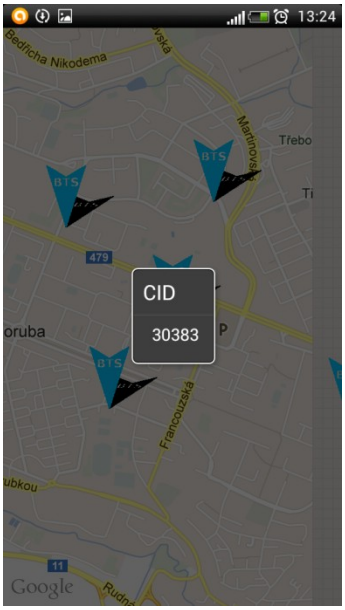
Obrázek 6 Hlavní obrazovka aplikace GSMInformator

Pokud se posunete v uživatelském rozhraní aplikace níže, můžete vidět parametry *Latitude* a *Longitude*. Ty představují GPS souřadnice. Pokud není GPS přijímač k dispozici, parametry mají hodnotu *Unknown*.

Poslední částí je navigační část se třemi tlačítky. Tlačítkem *ViewMap* můžete zobrazit mapu, na které budou zobrazeny dostupné BTS stanice. Pokud budou dostupné i GPS souřadnice, zobrazí se také poloha zařízení.

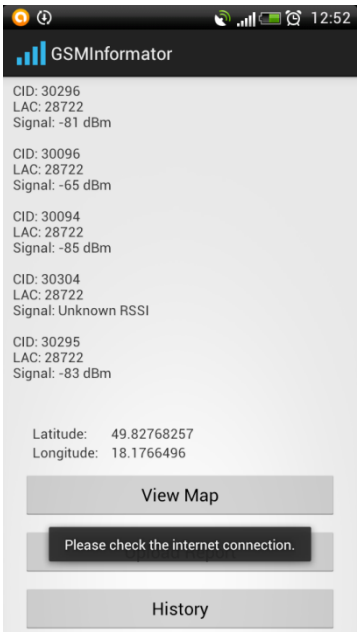
Obrázek 7 Zobrazení ikon na mapě

Jak je z obrázku č. 8 patrné, modré ikony zobrazují BTS stanice a zelená ikona zobrazuje polohu mobilního zařízení. Při kliknutí na jednotlivé ikony BTS se zobrazí jejich příslušné CID. Po kliknutí na ikonu polohy zařízení se zobrazí GPS souřadnice, které prozrazují aktuální geografickou polohu mobilního zařízení.



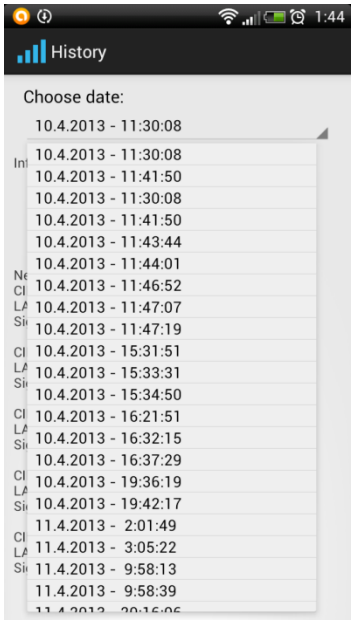
Obrázek 8 Zobrazení CID na mapě

Dalším tlačítkem *Upload Report* lze nahrát měřená data na server. K nahrání je potřeba přístup k internetu. Pokud je dostupné připojení k internetu, data se nahrají na server. Pokud připojení k internetu nebude k dispozici, budete na to upozorněni, jak můžete vidět na obrázku č. 10.



Obrázek 9 Upozornění o nedostupnosti připojení k internetu

Nahraná data lze zpětně prohlížet po stisknutí tlačítka *History*. Požadovaná data si můžete zobrazit výběrem data měření, které si můžete vybrat ve spinneru.

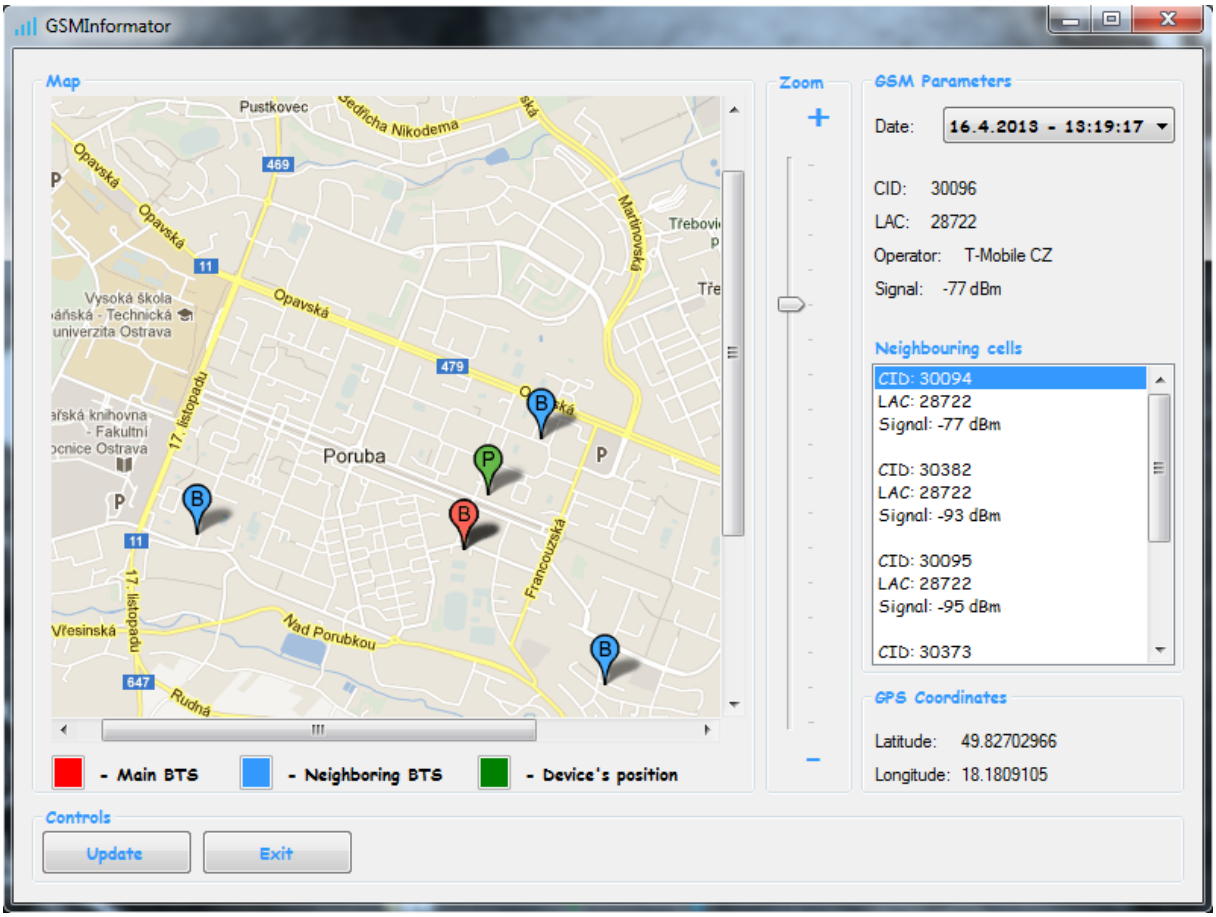


Obrázek 10 Výběr data pro zobrazení historie

Po výběru data se zobrazí měřené informace, které pak lze zobrazit na mapovém podkladu pomocí tlačítka *View Map*.

7.2 Aplikace pro Windows

Aplikace pro operační systém Windows slouží k zobrazování informací získaných měřením na mobilním zařízení.



Obrázek 11 GSMInformator pro operační system Windows

Největší část okna zabírá mapa, na které jsou zobrazeny BTS stanice a geografická poloha mobilního zařízení. Legenda pod mapou slouží k lepší orientaci na mapě. Červená ikona značí hlavní BTS stanici. Modrou barvou jsou znázorněny okolní BTS stanice a zelená barva určuje polohu mobilního zařízení. Napravo od mapy je oblast *Zoom*, která jak už název napovídá slouží k přiblížení a oddálení mapy. Toho lze docílit pomocí posuvníku. V části *GSM Parameters* jsou zobrazeny informace měřené na mobilním zařízením. Po spuštění aplikace se vždy zobrazí nejnovější informace, které uživatel nahrál z mobilního zařízení na server. Pro prohlížení informací měřených dříve si stačí vybrat požadované datum v seznamu *Date*. Část *GPS Coordinates* zobrazuje pouze GPS souřadnice, které byly získány mobilním zařízením.V navigační části *Controls* lze pomocí tlačítka *Update* aktualizovat seznam informací. Nejnovější měřená data lze pak zobrazit výběrem nejnovějšího data v seznamu *Date*. Tlačítko *Exit* slouží k ukončení aplikace.

8 Závěr

Cílem této bakalářské práce bylo vytvořit aplikaci pro mobilní zařízení s operačním systémem Android OS pro zjišťování informací o mobilní síti. Výsledky pak měly být zobrazeny na vzdáleném počítači.

Na základě tohoto cíle jsem vytvořil aplikaci *GSMInformator* pro operační systém Android, která reportuje parametry mobilní sítě GSM. Konkrétně se jedná o parametry *LAC, CID, síla signálu sítě* a *název operátor*. Dále ukládám *GSP souřadnice*, které jsou získány GPS přijímačem a *datum měření*, podle kterého lze později zobrazit výsledky z předchozích měření. Výsledky měření z aplikace se ukládají na paměťovou SD kartu mobilního zařízení, a také je lze odeslat na server. Součástí této bakalářské práce je také aplikace pro operační systém Windows, která slouží pro zobrazování měřených dat z mobilního zařízení.

Jelikož je ze strany Androidu špatná podpora pro reportování parametrů z okolních buněk v síti UMTS, je aplikace přizpůsobena pro sít GSM.

Aplikaci jsem testoval na mobilním zařízení v síti operátora T-Mobile. Seznam BTS stanic, který jsem si vytvořil, a podle kterého zobrazuji BTS stanice na mapovém podkladu, obsahuje pouze BTS stanice operátora T-Mobile. Vzhledem k četnosti BTS stanic na území České Republiky obsahuje seznam BTS stanic pouze BTS stanice na území Ostravy. Do budoucna plánuji rozšíření seznamu pro celé území České Republiky a pro všechny operátory v České Republice. Po zozšíření seznamu bude možno v aplikaci zobrazovat na mapovém podkladu všechny BTS stanice bez ohledu na mobilního operátora a geografickou polohu mobilního zařízení v rámci ČR.

Literatura

1. GUNNAR, Heine. *GSM Networks: Protocols, Terminology and Implementation*. Artech House, 1999. ISBN 0890064717, 9780890064719

2. EBERSPÄCHER, Jörg. *GSM - Architecture, Protocols and Services*. John Wiley & Sons, 2008. ISBN 0470741724, 9780470741726

3. GWENAËL, Le Bodic. *Mobile Messaging Technologies and Services: SMS, EMS and MMS*. John Wiley & Sons, 2005. ISBN 0470014512, 9780470014516

4. UJBÁNYAI, Miroslav. *Programujeme pro Android*. Grada Publishing a.s., 2012. ISBN 802473995X, 9788024739953

5. Android Developers: Activities. [online]. [cit. 2013-04-24]. Dostupné z: http://developer.android.com/guide/components/activities.html

6. Android Developers: Services. [online]. [cit. 2013-04-24]. Dostupné z: http://developer.android.com/guide/components/services.html

7. Android Developers: Content Providers. [online]. [cit. 2013-04-24]. Dostupné z: http://developer.android.com/guide/topics/providers/content-providers.html

8. Android founder: We aimed to make a camera OS. [online]. [cit. 2013-04-25]. Dostupné z: http://www.pcworld.com/article/2034723/android-founder-we-aimed-to-make-a-camera-os.html

9. Android slaví pět let a 75% podíl na trhu. [online]. [cit. 2013-04-23]. Dostupné z: http://www.linuxexpres.cz/android-slavi-pet-let-a-75-podil-na-trhu

10. Android Developers. [online]. [cit. 2013-04-24]. Dostupné z: http://developer.android.com/about/versions/index.html

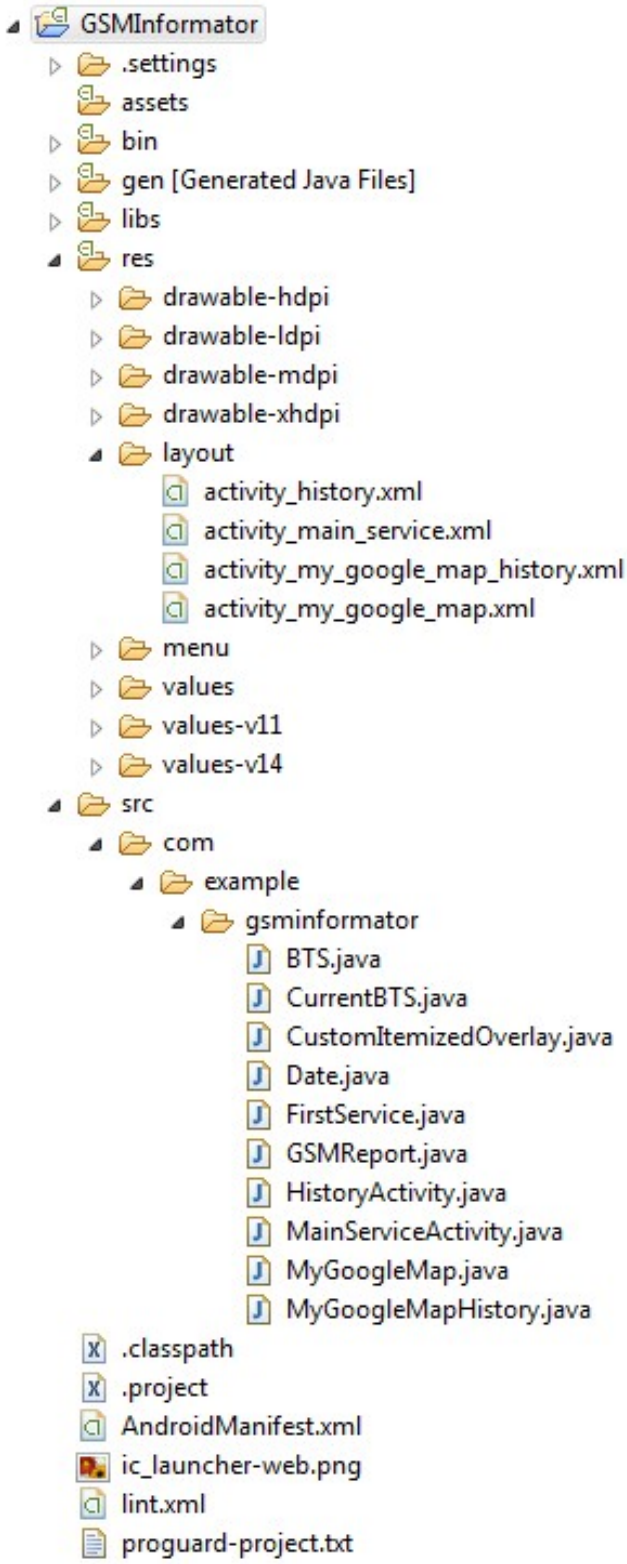
11. VOGEL, Lars. Android BroadcastReceiver Tutorial. [online]. [cit. 2013-04-27]. Dostupné z: http://www.vogella.com/articles/AndroidBroadcastReceiver/article.html

12. RICHTR, Tomáš. Technologie pro mobilní komunikaci. [online]. [cit. 2013-04-28]. Dostupné z: http://tomas.richtr.cz/mobil/index.htm

13. MICHALEK, Libor. Telekomunikační sítě – Mobilní rádiové sítě. (přednáška) Ostrava: VŠB-TUO, 2.5.2011

14. *Android Radio Layer Interface* [online]. [cit. 2013-04-30]. Dostupné z: http://www.slideshare.net/ssusere3af56/android-radio-layer-interface

A Adresářová struktura aplikace GSMInformator



B Adresářová struktura přiloženého DVD

Android/	Obsahuje kompletní projekt vytvořené aplikace pro operační system Android
Windows/	Obsahuje kompletní projekt vytvořené aplikace pro operační system Windows
Texty/	Obsahuje text bakalářské práce ve formátu *.pdf
Obrázky/	Obsahuje veškeré obrázky použité v této bakalářské práci